Structural Cryptanalysis of McEliece Schemes with Compact Keys

Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, Frédéric de Portzamparc, Jean-Pierre Tillich

Structural Cryptanalysis of McEliece Schemes with Compact Keys

Jean-Charles Faugère[1,2,3], Ayoub Otmani[4], Ludovic Perret[2,1,3] Frédéric de Portzamparc [6,1,2,3] and Jean-Pierre Tillich[5]

Sorbonne Universités, UPMC Univ Paris 06, POLSYS, UMR 7606, LIP6, F-75005, Paris, France[1]
INRIA, Paris-Rocquencourt Center[2],
CNRS, UMR 7606, LIP6, F-75005, Paris, France[3] Normandie Univ, France; UR, LITIS, F-76821 Mont-Saint-Aignan, France.[4]
INRIA, Paris-Rocquencourt Center[5],
Gemalto, 6 rue de la Verrerie 92190, Meudon, France[6]
jean-charles.faugere@inria.fr,ayoub.otmani@univ-rouen.fr,ludovic.perret@lip6.fr,
frederic.urvoydeportzamparc@gemalto.com,jean-pierre.tillich@inria.fr

Abstract. A very popular trend in code-based cryptography is to decrease the public-key size by focusing on subclasses of alternant/Goppa codes which admit a very compact public matrix, typically quasi-cyclic (QC), quasi-dyadic (QD), or quasi-monoidic (QM) matrices. We show that the very same reason which allows to construct a compact public-key makes the key-recovery problem intrinsically much easier. The gain on the public-key size induces an important security drop, which is as large as the compression factor p on the public-key. The fundamental remark is that from the $k \times n$ public generator matrix of a compact McEliece, one can construct a $k/p \times n/p$ generator matrix which is – from an attacker point of view – as good as the initial public-key. We call this new smaller code the *folded code*. Any key-recovery attack can be deployed equivalently on this smaller generator matrix. To mount the key-recovery in practice, we also improve the algebraic technique of Faugère, Otmani, Perret and Tillich (FOPT). In particular, we introduce new algebraic equations allowing to include codes defined over any prime field in the scope of our attack. We describe a so-called "structural elimination" which is a new algebraic manipulation which simplifies the key-recovery system. As a proof of concept, we report successful attacks on many cryptographic parameters available in the literature. All the parameters of CFS-signatures based on QD/QM codes that have been proposed can be broken by this approach. In most cases, our attack takes few seconds (the harder case requires less than 2 hours). In the encryption case, the algebraic systems are harder to solve in practice. Still, our attack succeeds against several cryptographic challenges proposed for QD and QM encryption schemes, but there are still some parameters that have been proposed which are out of reach of the methods given here. However, regardless of the key-recovery attack used against the folded code, there is an inherent weakness arising from Goppa codes with QM or QD symmetries. It is possible to derive from the public key a much smaller public key corresponding to the folding of the original QM or QD code, where the reduction factor of the code length is precisely the order of the QM or QD group used for reducing the key size. To summarize, the security of such schemes are not relying on the bigger compact public matrix but on the small folded code which can be efficiently broken in practice with an algebraic attack for a large set of parameters.

Keywords. public-key cryptography, McEliece cryptosystem, algebraic cryptanalysis, folded code, 11T71

1 Introduction

The McEliece cryptosystem, which was presented in the late seventies [34], still belongs to the very few public key cryptosystems that remain unbroken. Despite its impressive resistance against

a variety of attacks and its fast encryption and decryption, McEliece has not (yet) been really deployed in practical applications. This is most likely due to the large size of the public-key. To overcome this limitation, a very popular research trend is to decrease the public-key size by focusing on subclasses of alternant/Goppa codes which admit a very compact parity-check or generator matrix [25,8,35,3,38]. This reduction is obtained by taking classes of such codes which have *quasi-cyclic* (QC) or *quasi-dyadic* (QD) generator matrices. The hope is that the additional structure does not deteriorate the security of the system. This is very much in the spirit of using ideal lattices instead of standard lattices in lattice-based cryptography [41,31].

This hope was eroded by the algebraic attacks presented in [22,23,43]. Algebraic cryptanalysis is a general framework that permits to assess the security of a large variety of cryptographic schemes. In [22,23], Faugère, Otmani, Perret and Tillich (FOPT) extended the scope of algebraic attacks to McEliece-like (i.e. using alternant or Goppa codes) cryptosystems, targeting in particular two compact variants of McEliece [8,35]. Whilst most previous security analysis of McEliece focused on improving general decoding algorithms, e.g. [28,29,42,15,9,12,33,4], FOPT is one of the rare general techniques – with the support splitting algorithm [40,30] – focusing on key-recovery and trying to exploit the structure of Goppa/alternant codes. More specifically, [22] shows that the secret-key in McEliece-like cryptosystems can be recovered by solving the following system:

$$\left\{ g_{i,0} Y_0 X_0^\ell + \cdots + g_{i,n-1} Y_{n-1} X_{n-1}^\ell = 0 \;\mid\; i \in \{0,\ldots,k-1\}, \ell \in \{0,\ldots,t-1\} \right\} \tag{1}$$

where the unknowns $\boldsymbol{X} \stackrel{\text{def}}{=} (X_0,\ldots,X_{n-1})$ and $\boldsymbol{Y} \stackrel{\text{def}}{=} (Y_0,\ldots,Y_{n-1})$ correspond to the secret support and multiplier respectively, and the $g_{i,j}$'s are the entries of the public matrix. However, for the original parameters proposed by McEliece (e.g. $n = 1024$ and $t = 50$), the algebraic system (1) has 2048 variables and equations of degree up to 49. Today, this is clearly beyond classical Gröbner bases algorithms such as [14,17,18]. On the other hand, for compact variants of McEliece using QC codes [8], or QD codes [35], the situation is different. Indeed, the system (1) can be drastically simplified, allowing to break all the parameters proposed for QC codes [8] as well as many QD challenges [8].

Despite this preliminary cryptanalytic result, the design of compact McEliece schemes remains a rather popular topic of research e.g. [27,3,38,1]. This can be explained by the fact that [22,23] pointed that *binary* QD codes seem to resist the initial FOPT attack. This motivated then Barreto and Misoczki to revise their initial parameters and recommend to use only QD binary codes [36]. This also yielded a series of papers implementing efficiently QD binary codes [27], proposing new uses of such codes ([2] describes a QD version of the CFS [24] signature scheme) and new compact subclasses of alternant codes (Persichetti [38] proposes quasi-dyadic Srivastava codes), and even generalizing QD codes: [3] introduces *quasi-monoidic* (QM) codes. As pointed in [3], QM codes lead to even smaller parameters than QD codes. The idea underlying QM codes is that one can enhance the security of binary QD codes by considering prime fields of sizes $q > 2$.

1.1 How to Use Symmetries for Breaking (Some) Compact Variants of McEliece

Structural Weakness of QD *and* QM *Schemes.* In [22], the authors use the very structure of QD or QC codes [8,35] to obtain linear relations on supports and multipliers. These linear dependencies allow to reduce the number of unknowns without changing the degree of the system (1). After this simplification, the new system has a shape similar to (1) but with a much smaller number of unknowns. This suggests that there might be a hidden Goppa (resp. alternant) code behind such

QD Goppa (resp. QC alternant) codes but with much smaller parameters. A major result of this paper is to show in the QM context that this intuition is valid. This is the object of Section 3. We provide an explicit way to construct the smaller "hidden" code from the public generator matrix. Let $\boldsymbol{G} \in \mathbb{F}_q^{k \times n}$ be the public key of a QM scheme. Assume that p is the compression factor of the compact public-key (compared to a plain McEliece). We show in Theorem 3 that it possible to construct from the public-key $\boldsymbol{G}$ a $k/p \times n/p$ generator matrix of a new Goppa code. We call this new code the *folded Goppa code*. In Sec. 3.2, we show that the secret algebraic structure of the original Goppa code (i.e. its support and Goppa polynomial) used during the decryption process can be easily recovered from the support and the Goppa polynomial of the folded Goppa code.
This implies that a key-recovery on QD and QM schemes is not harder than a key-recovery on a reduced McEliece scheme where all parameters have been scaled down by a factor of p, which is the compression factor allowed by the QD or QM structure. For instance, we can reduce the key-recovery of a QD Goppa code of length 8192 and dimension 4096 (parameters suggested in [35]) to the key-recovery on a QD Goppa code of length 64 and dimension 32. In other words, the very reason which allowed to design compact variants of McEliece can be used to attack such schemes much more efficiently. We also point out that the results of Section 3 can be further generalized. It was shown in [8,5,7,6] that QD, QM and QC alternant codes can be unified through a common framework. Indeed, all these codes are constructed in such a way as to have a non-trivial permutation. We can then generalize Theorem 3 as well as the folding process to a large variety of codes with a non-trivial permutation. This allows to prove that, for all known compact variants of McEliece, we can reduce the key-recovery on the initial compact code to a much smaller code which has eventually no more symmetry (i.e. a plain Goppa code). For key-recovery, this implies that the hardness of a compact variant of McEliece is equivalent to the security of a standard McEliece with scaled-down parameters. We present in this paper some theoretic results allowing to attack QD/QM schemes, as well as the tools which are necessary to mount the attack in practice. A general theoretical treatment of alternant codes with non-trivial automorphism groups and their folding will be the purpose of a forthcoming paper [**?**].

Improvements of the Algebraic Modelling. The folding process described before holds regardless of the key-recovery method chosen. Although the folded code have rather small parameters, the support splitting attack [40,30] will not be efficient at all in this context. The reason is that the codes obtained are not full support. This induces another (important) combinatorial factor to the enumeration of the Goppa polynomials. For this reason, we use the algebraic attack [22] to effectively mount the key-recovery attack. Along the way, this required to develop new algebraic tools. In Section 4, we define precisely the set of algebraic equations which can be generated in function of the type of code considered: namely alternant, Goppa, or binary Goppa codes. System (1) only describes an alternant decoder, whilst more equations can be added for Goppa and binary Goppa codes. For alternant codes, we briefly recall in Sec. 4.1 the basic ideas of FOPT [22,23], i.e. we explain how the system (1) is constructed. For (binary) Goppa codes, we show in Sec. 4.2 that new additional equations can be generated by using the Goppa polynomial. This addresses a question reported by several authors (e.g. [2,1]), i.e. how to describe completely a Goppa (and binary Goppa) decoder with the algebraic attack of [22]. In Sec. 4.3, we present a novel technique, so-called *structured elimination*, allowing to simplify (i.e. by eliminating some variables) the algebraic system corresponding to any alternant code. Note that elimination is a classical task which can be done by computing a Gröbner basis with a suitable order [16, Chap. 3]. Here, we use the structure of the system to eliminate variables without any Gröbner basis computation. Our elimination process

is much more efficient than the generic method [16, Chap. 3]. We emphasize that all the results presented in Section 4 do not use the symmetries of QD or QM codes. The results of Section 4 are then an improved and refined suite of algebraic tools for attacking any McEliece-like cryptosystem. Section 5 explains how to combine, for QD and QM codes, the folding technique from Section 3 with the algebraic equations as well as the structural elimination of Section 4. We present several experimental results on the set of parameters proposed in [35,2,3]. For signature schemes based on QD/QM codes, our attack is particularly efficient. In this case, it is well known that the codes which can be used in this context have necessarily a very small redundancy. The number of unknowns which remain after structured elimination is in this case much smaller than the number of equations. Consequently, all the parameters suggested for QD-CFS [2] can be broken in a few seconds (Table 2). For QM-CFS [3], a parameter requires less than 2 hours, but all the others can be broken in a few seconds (Table 2). In view of our new attack, it seems extremely hard to find parameters of cryptographic interest for friendly-CFS QD/QM codes. In the encryption case, the algebraic systems are harder to solve in practice. Still, we report several successful results against challenges proposed for QD/QM encryption schemes. To measure the progresses realized in comparison to [22], we report below some practical results obtained with our new techniques and some results from [22,23](the notation N.A. means that the parameters could not be addressed in FOPT). The results of [22,23] were obtained with the FGB software[19]; an optimized C implementation of the F_5 algorithm [18]). It is interesting to see that in the non-binary case, our attack can be easily reproduced using on-the-shelf computer algebra system MAGMA [13]. The huge speed-up that can be readily observed is due to the improved modelling presented, and vastly thanks to the folding process.

$q=2$	m	t	This paper using MAGMA	This paper using F_5/FGB	FOPT [22]	Sec. level
2	16	32	18 s.		N.A.	128
2	12	128		$\leqslant 2^{83.5}$ op.	N.A.	128
2	14	128		$\leqslant 2^{96.1}$ op.	N.A.	226
2	15	512		$\leqslant 2^{146}$ op.	N.A.	256
2	16	256		$\leqslant 2^{168}$ op.	N.A.	218
2	16	256		$\leqslant 2^{157}$ op.	N.A.	256
2^4	4	64	0.010 s.		0.50 s	128
2^4	4	128	0.010 s.		7.1 s	128
2^2	8	64	0.040 s.		1,776.3 s	128

Unlike, [22,23], it appears that the underlying characteristic of the field does not really influence the complexity of our attack. Typically, we can mount our attack in the binary case. We also mention that some parameters proposed could not be solved in practice.

2 Coding Theory Background

Let $\mathbb{F}_q$ be a finite field of $q = p^s$ elements (p prime, and $s > 0$). A linear *code* $\mathscr{C}$ of *length* n and *dimension* k over $\mathbb{F}_q$ is a subspace of dimension k of the full space $\mathbb{F}_q^n$. It can be specified by a full-rank matrix called a *generator matrix* which is a $k \times n$ matrix $\boldsymbol{G}$ (with $k \leqslant n$) over $\mathbb{F}_q$ whose rows span the code, namely $\mathscr{C} \stackrel{\text{def}}{=} \left\{\boldsymbol{uG} \mid \boldsymbol{u} \in \mathbb{F}_q^k\right\}$. It can also be defined as the right kernel of a *parity-check* matrix $\boldsymbol{H}$, that is $\mathscr{C} = \left\{\boldsymbol{x} \in \mathbb{F}_q^n \mid \boldsymbol{Hx}^T = \boldsymbol{0}\right\}$, where $\boldsymbol{x}^T$ is the transpose of the row vector $\boldsymbol{x}$. The McEliece cryptosystem relies on binary Goppa codes, which belong to the class of *alternant codes*. It is convenient to describe this latter class through a parity-check matrix over an

extension field $\mathbb{F}_{q^m}$ of $\mathbb{F}_q$ over which the code is defined. For alternant codes of length $n \leqslant q^m$, there exists a parity-check matrix with a very special form:

$$\boldsymbol{V}_t(\boldsymbol{x}, \boldsymbol{y}) \stackrel{\text{def}}{=} \begin{pmatrix} y_0 & \cdots & y_{n-1} \\ y_0 x_0 & \cdots & y_{n-1} x_{n-1} \\ \vdots & & \vdots \\ y_0 x_0^{t-1} & \cdots & y_{n-1} x_{n-1}^{t-1} \end{pmatrix}, \tag{2}$$

where $\big(\boldsymbol{x} = (x_0, \ldots, x_{n-1}), \boldsymbol{y} = (y_0, \ldots, y_{n-1})\big) \in \mathbb{F}_{q^m}^n \times \mathbb{F}_{q^m}^n$.

Definition 1 (Alternant code). *Let $\boldsymbol{x} = (x_0, \ldots, x_{n-1}) \in (\mathbb{F}_{q^m})^n$ where all x_i's are distinct and $\boldsymbol{y} = (y_0, \ldots, y_{n-1}) \in \left(\mathbb{F}_{q^m}^*\right)^n$. The alternant code of order t over $\mathbb{F}_q$, associated to $\boldsymbol{x}$ and $\boldsymbol{y}$, is $\mathscr{A}_t(\boldsymbol{x}, \boldsymbol{y}) \stackrel{def}{=} \left\{\boldsymbol{c} \in \mathbb{F}_q^n \mid \boldsymbol{V}_t(\boldsymbol{x}, \boldsymbol{y})\boldsymbol{c}^T = \boldsymbol{0}\right\}$. The dimension k satisfies $k \geqslant n - tm$. We shall call here $\boldsymbol{x}$ the* support *of the code, $\boldsymbol{y}$ the* multiplier *and t the* order *(or* degree*) of the alternant code.*

A key feature about alternant codes of degree t is the fact that there exists a polynomial time algorithm decoding all errors of weight at most $\frac{t}{2}$ *once a parity-check matrix is given in the form* $\boldsymbol{V}_t(\boldsymbol{x}, \boldsymbol{y})$ [32, Ch.12, 9]. For subclasses of alternant codes, algorithms correcting more errors can be found. A widely used example is the one of *binary Goppa codes.*

Definition 2 (Goppa codes). *The Goppa code $\mathscr{G}(\boldsymbol{x}, \Gamma)$ over $\mathbb{F}_q$ associated to a polynomial $\Gamma(z) \in \mathbb{F}_{q^m}[z]$ of degree t and n-tuple $\boldsymbol{x} = (x_0, \ldots, x_{n-1})$ of distinct elements of $\mathbb{F}_{q^m}$ satisfying $\Gamma(x_i) \neq 0$ for all $i, 0 \leqslant i \leqslant n-1$, is the alternant code $\mathscr{A}_t(\boldsymbol{x}, \boldsymbol{y})$ of order t with $y_i = \Gamma(x_i)^{-1}$ for all $i, 0 \leqslant i \leqslant n-1$. Equivalently, $\mathscr{G}(\boldsymbol{x}, \Gamma)$ is defined as the set of codewords $\boldsymbol{c} = (c_0, \ldots, c_{n-1}) \in \mathbb{F}_q^n$ such that:*

$$P_{\boldsymbol{c},\boldsymbol{x}}(z) \stackrel{def}{=} \sum_{i=0}^{n-1} \frac{c_i}{z - x_i} \equiv 0 \bmod \Gamma(z). \tag{3}$$

We shall call $P_{\boldsymbol{c},\boldsymbol{x}}(z)$ the syndrome polynomial associated to $\boldsymbol{c} \in \mathscr{G}(\boldsymbol{x}, \Gamma)$.

Goppa codes, viewed as alternant codes, naturally inherit a decoding algorithm that corrects up to $\frac{t}{2}$ errors. For *binary* Goppa codes, we can improve this bound to correct twice as many errors.

Theorem 1. *[32, p. 341], [37] Let $\Gamma(z)$ be a polynomial of degree t without multiple roots. The binary Goppa code $\mathscr{G}(\boldsymbol{x}, \Gamma)$ is equal to the alternant code $\mathscr{A}_{2t}(\boldsymbol{x}, \boldsymbol{y})$, with $y_i = \Gamma(x_i)^{-2}$ for all $i, 0 \leqslant i \leqslant n-1$. As a consequence, there exists a polynomial time algorithm decoding all errors of Hamming weight at most t in $\mathscr{G}(\boldsymbol{x}, \Gamma)$ as soon as $\boldsymbol{x}$ and $\Gamma(z)$ are known.*

3 Quasi-Monoidic Codes Revisited

The purpose of this section is twofold. First, we describe (Theorem 2) a more general construction of QM Goppa codes than in [35,2,3]. This allows for instance to obtain QM Goppa codes which are irreducible, something which is out of reach of the methods proposed in [35,2,3]. In addition, this result permits to have, for instance, more flexibility in choosing the parameters of the scheme. Then, we propose an operation on codes having a non-trivial permutation group. This operation allows to scale-down the parameters of the code. We call *folded code* the resulting reduced code (Definition 5). QM Goppa codes obtained from the aforementioned construction, i.e. Theorem 2, produces Goppa codes which have a non-trivial permutation group. Thus, we can apply the folding procedure to QM codes. The folding process yields a way of producing from a QM public-key a new public-key whose parameters have been reduced by the order of the QM permutation group (Theorem 3).

Definition 3 (Quasi-monoidic and quasi-dyadic codes [3,35]). *Let $q = p^s$ (p prime, and $s > 0$). Let λ be an integer such that $p^\lambda \leqslant q$. A matrix $\boldsymbol{H} = (h_{i,j}) \in \mathbb{F}_q^{p^\lambda \times p^\lambda}$ is* monoidic *if $\forall i, j, 0 \leqslant i, j < p^\lambda$: $h_{i,j} = h_{0,j \ominus i}$, where for any integers $a \geqslant 0$ and $b \geqslant 0$ the operations $a \oplus b$ and $a \ominus b$ stand for component-wise addition and subtraction modulo p of their p-ary decomposition. When $p = 2$, a monoidic matrix is called a* dyadic *matrix. A* quasi-monoidic *(resp.* quasi-dyadic*) matrix is a block-matrix such that each block is monoidic (resp. dyadic). A* quasi-monoidic *(*QM*) code is a linear code which admits a quasi-monoidic parity-check matrix.* Quasi-dyadic *(*QD*) codes are defined in a similar way.*

The authors of [3,35] address the question of constructing QD and QM Goppa codes with separable Goppa polynomials. They use the fact that if the Goppa polynomial is separable and has single roots then the associated Goppa code admits a parity-check matrix in *Cauchy form* $C(\boldsymbol{x}, \boldsymbol{z}) \stackrel{\text{def}}{=} \left[\frac{1}{z_i - x_j}\right]_{\substack{0 \leqslant i \leqslant t-1 \\ 0 \leqslant j \leqslant n-1}}$ where $\boldsymbol{x}$ is the support of the code and $\boldsymbol{z}$ is the set of the roots of the Goppa polynomial (See [32][Ch. 12, 3, p.345]). Their idea is to find $\boldsymbol{x}$ and $\boldsymbol{z}$ such as $C(\boldsymbol{x}, \boldsymbol{z})$ is also monoidic (or dyadic when $p = 2$). In [3, Theorem 1] and [35, Theorem 2], the authors provide a precise characterisation of the constraints that $\boldsymbol{x}$ and $\boldsymbol{z}$ should satisfy, in particular the block size has to be a power of the characteristic p of $\mathbb{F}_q$. This leads to algorithm [3, Algo. 3] generating monoidic and quasi-monoidic parity-check matrices.

3.1 Linear Codes with Non Trivial Automorphism Group

We define below an object of particular interest for this paper.

Definition 4 (Automorphism group of a code restricted to permutations). *Let $\boldsymbol{c} \in \mathscr{C}$ and σ be a permutation of the code positions (i.e. a permutation on the indices $\{0, \ldots, n-1\}$). The permutation σ acts on a codeword as follows $\boldsymbol{c}^\sigma = (c_{\sigma(i)})_{0 \leqslant i \leqslant n-1}$. We shall say that σ is an* automorphism *of $\mathscr{C}$ if and only if $\boldsymbol{c}^\sigma \in \mathscr{C}$, for all $\boldsymbol{c} \in \mathscr{C}$. Finally, we denote by* $\mathrm{Aut}(\mathscr{C})$ *the set of automorphisms (restricted to permutations on the code positions) of $\mathscr{C}$.*

We first observe the fact that QM codes have a non-trivial automorphism group.

Proposition 1. *Let $t = t_0 p^\lambda$, and $\mathscr{C}$ be a* QM *code with a parity-check matrix $\boldsymbol{H} \in \mathbb{F}_{q^m}^{t \times n}$ with monoidic blocks of size $p^\lambda \times p^\lambda$. For $\ell, 0 \leqslant \ell < p^\lambda$, let σ_ℓ be the permutation acting on $\{0, 1, \ldots, n-1\}$ as $\sigma_\ell(i) = i \ominus \ell$. Then, $\forall \ell, 0 \leqslant \ell < p^\lambda$, the permutation defined by σ_ℓ is an automorphism of $\mathscr{C}$. Conversely, a code $\mathscr{C}$ of length $n = n_0 p^\lambda$ such that for all $\ell, 0 \leqslant \ell < p^\lambda, \sigma_\ell \in \mathrm{Aut}(\mathscr{C})$, then such a code has a* QM *parity-check matrix.*

Proof. By definition, $\mathscr{C}$ admits a parity-check matrix made of blocks of size $p^\lambda \times p^\lambda$. This implies that the length n of $\mathscr{C}$ is of the form $n = n_0 p^\lambda$. As a consequence, it is not hard to see that the set of the indices $\{0, \ldots, n-1\}$ is globally invariant by all the permutations $\sigma_\ell : j \mapsto j \ominus \ell$ for $0 \leqslant \ell \leqslant p^\lambda - 1$. Let us first assume that $t_0 = 1$. Let $\boldsymbol{c} \in \mathscr{C}$. We want to show that $\boldsymbol{c}^{\sigma_\ell} = (c_{j \ominus \ell})_{0 \leqslant j < n} \in \mathscr{C}$. Since $\boldsymbol{c} \in \mathscr{C}$ and $\boldsymbol{H}$ is quasi-monoidic, it holds that $0 = \sum_{j=0}^{n-1} c_j H_{i,j} = \sum_{j=0}^{n-1} c_j H_{0, j \ominus i}$, for all $i, 0 \leqslant i < t = p^\lambda$. We then make a change of indices $j' = j \ominus \ell$ in the syndrome polynomial associated to $\boldsymbol{c}^{\sigma_\ell}$:

$$\sum_{j=0}^{n-1} c_{j \ominus \ell} H_{0, j \ominus i} = \sum_{j'=0}^{n-1} c_{j'} H_{0, (j' \oplus \ell) \ominus i)} = \sum_{j'=0}^{n-1} c_{j'} H_{0, j' \ominus (i \ominus \ell)} = \sum_{j'=0}^{n-1} c_{j'} H_{0, j' \ominus i'}.$$

where $i' = i \ominus \ell$ satisfies $0 \leqslant i' < p^\lambda$. Therefore the last sum is equal to zero. This implies that $\boldsymbol{c}^{\sigma_\ell} \in \mathscr{C}$ and that σ_ℓ is an automorphism of $\mathscr{C}$. When $t_0 > 1$, the code $\mathscr{C}$ is the intersection of the codes $\mathscr{C}_i$ with $0 \leqslant i < t_0$ where $\mathscr{C}_i$ is a code with a QM parity check matrix formed by the rows of $\boldsymbol{H}$ of index s in the range $ip^\lambda \leqslant s < (i+1)p^\lambda$. σ_ℓ is an automorphism of all these codes (this is precisely what we have already proved) and therefore σ_ℓ is also an automorphism of their intersection $\mathscr{C}$. □

Remark 1. Let $\ell, 0 \leqslant \ell \leqslant p^\lambda - 1$. The permutations σ_ℓ defined in Proposition 1 have a block-wise action on the indices. This is more explicit with the formulation $\sigma_\ell : i \mapsto \lfloor i/p^\lambda \rfloor p^\lambda + \big((i \mod p^\lambda) \ominus \ell\big)$.

T. Berger proved in [5] that alternant codes with non-trivial permutation group can be designed by imposing relations to their support and multipliers. In [35, Algorithm 1] and [3, Algorithm 3] the authors describe methods to sample efficiently QD Goppa codes and QM codes respectively. Compared to [5], the codes designed in [35,3] have the interesting property of being invariant by several permutations simultaneously, thus leading to larger permutation groups. However, as pointed out by these authors, the algorithms only permit to sample a sub-space of QD/QM codes. The next result allows – in particular – to generate a larger portion of such compact codes. Of greater interest for this paper, this characterisation allows to reveal a structural weakness of QD/QM codes. The next result can be seen as a specialization/generalization to our context of the work of T. Berger [8,5,7,6] on codes with non-trivial permutation groups. The proof of this result is provided for the sake of completeness.

Theorem 2. *Let* $\mathrm{char}(\mathbb{F}_{q^m}) = p$*, and let* $\gamma(z) \in \mathbb{F}_{q^m}[z]$ *be of degree* t_0*. Let* $\alpha_0, \ldots, \alpha_{\lambda-1} \in \mathbb{F}_{q^m}$ *be a set of* λ *elements which are* $\mathbb{F}_p$*-independent over* $\mathbb{F}_{q^m}$*. We denote by* $G \subset \mathbb{F}_{q^m}$ *the additive group of the* $\mathbb{F}_p$*-linear combinations of the* α_i*'s. Let* $\widetilde{x}_0, \widetilde{x}_1, \ldots, \widetilde{x}_{(n_0-1)}$ *be elements of* $\mathbb{F}_{q^m}$ *which are in different cosets of* $\mathbb{F}_{q^m}/G$*. Let* $(i_0, \ldots, i_{\lambda-1}) \in \mathbb{F}_p^\lambda$ *be the representation of* $i \bmod p^\lambda$ *in base* p*, i.e.* $i \equiv \sum_{j=0}^{\lambda-1} i_j p^j \bmod p^\lambda$*. Let* $n \stackrel{def}{=} n_0 p^\lambda$*. We define* $\boldsymbol{x} = (x_i)_{0 \leqslant i < n}$ *and* $\Gamma(z)$ *as follows:*

$$x_i = \widetilde{x}_{\lfloor i/p^\lambda \rfloor} + \sum_{j=0}^{\lambda-1} i_j \alpha_j, \tag{4}$$

$$\Gamma(z) = \gamma\big(P(z)\big) \text{ where } P(z) = \textstyle\prod_{g \in G}(z - g). \tag{5}$$

Then, $\mathscr{G}(\boldsymbol{x}, \Gamma(z))$ *is a Goppa code of length* n*, degree* $t = t_0 p^\lambda$ *which admits an automorphism group of size* p^λ*.*

Before proving Theorem 2, we show first that imposing certain constraints on a Goppa code allows to design codes with a chosen permutation.

Proposition 2. *Let* $\mathscr{G}(\boldsymbol{x}, \Gamma)$ *be a Goppa code of order* t*. Let* $\ell, 0 \leqslant \ell \leqslant t-1$ *be an integer such that* σ_ℓ *defined by* $\sigma_\ell(i) \stackrel{def}{=} i \ominus \ell$ *induces a permutation of the code positions. We assume that there exists* $g_\ell \in \mathbb{F}_{q^m}$ *such that, for all* $i, 0 \leqslant i \leqslant n-1, x_{i \oplus \ell} = x_i + g_\ell$ *and* $\Gamma(z - g_\ell) = \Gamma(z)$*. Then, for all* $\ell, 0 \leqslant \ell \leqslant t-1$*, the code coordinate permutation* σ_ℓ *is an automorphism of* $\mathscr{G}(\boldsymbol{x}, \Gamma)$*.*

Proof. Let $\boldsymbol{c} \in \mathscr{G}(\boldsymbol{x}, \Gamma)$. The syndrome polynomial $P_{\boldsymbol{c}^{\sigma_\ell}, \boldsymbol{x}}$ associated to $\boldsymbol{c}^{\sigma_\ell}$ verifies

$$P_{\boldsymbol{c}^{\sigma_\ell}, \boldsymbol{x}}(z) = \sum_{i=0}^{n-1} \frac{c_{\sigma_\ell(i)}}{z - x_i} = \sum_{i=0}^{n-1} \frac{c_{i \ominus \ell}}{z - x_i} = \sum_{i=0}^{n-1} \frac{c_i}{z - x_{i \oplus \ell}} = \sum_{i=0}^{n-1} \frac{c_i}{z - g_\ell - x_i} = P_{\boldsymbol{c}, \boldsymbol{x}}(z - g_\ell).$$

According to Definition 2, we have $\Gamma(z)|P_{\boldsymbol{c},\boldsymbol{x}}(z)$. So, $\Gamma(z) = \Gamma(z-g_\ell)|P_{\boldsymbol{c}^{\sigma_\ell},\boldsymbol{x}}(z)$ and $\boldsymbol{c}^{\sigma_\ell} \in \mathscr{G}(\boldsymbol{x},\Gamma)$. □

Thanks to Proposition 2, we are now in a position to explain why the supports and polynomials given in Theorem 2 produce codes with admit an automorphism group isomorphic to $(\mathbb{Z}/p\mathbb{Z})^\lambda$.

Proof. Let us show that all the permutations $\sigma_\ell(i) = i \ominus \ell$, for $\ell, 0 \leqslant \ell \leqslant p^\lambda - 1$ are code permutations. We start by proving it for $\ell = p^u$, with $u, 0 \leqslant u \leqslant \lambda - 1$. For $i, 0 \leqslant i \leqslant t_0 p^\lambda - 1$, let $(i_0, \ldots, i_{\lambda-1}) \in \mathbb{F}_p^\lambda$ be such that $i \equiv \sum_{j=0}^{\lambda-1} i_j p^j \bmod p^\lambda$. The representation $(i'_0, \ldots, i'_{\lambda-1})$ of $(i \oplus p^u) \bmod p^\lambda$ in base p is deduced from the previous representation by

$$\begin{aligned} i'_j &= i_j, \quad \text{if } j \neq u, \\ i'_u &\equiv i_u + 1 \bmod p \text{ otherwise.} \end{aligned}$$

Hence, $i \oplus p^u$ and i only differ in their u-th digit. Since u is smaller than λ, we get:

$$\lfloor (i \oplus p^u)/p^\lambda \rfloor p^\lambda = \lfloor i/p^\lambda \rfloor p^\lambda \,.$$

Hence, thanks to Equation (4) we can write:

$$\begin{aligned} x_{i \oplus p^u} &= \widetilde{x}_{\lfloor (i \oplus p^u)/p^\lambda \rfloor p^\lambda} + \sum_{j=0}^{\lambda-1} i'_j \alpha_j = \widetilde{x}_{\lfloor i/p^\lambda \rfloor p^\lambda} + \alpha_u + \sum_{j=0}^{\lambda-1} i_j \alpha_j \\ &= x_i + \alpha_u. \end{aligned} \tag{6}$$

Now pick any ℓ in $[0, \ldots, p^\lambda - 1]$ and decompose it in base p, that is: $\ell = \sum_{j=0}^{\lambda-1} \ell_j p^j$. A quick induction on Equation (6) shows that for all $i, 0 \leqslant i \leqslant t_0 p^\lambda - 1$:

$$\begin{aligned} x_{i \oplus \ell} &= x_i + \textstyle\sum_{j=0}^{\lambda-1} \ell_j \alpha_j \\ &= x_i + g_\ell \end{aligned}$$

where we set $g_\ell = \sum_{j=0}^{\lambda-1} \ell_j \alpha_j \in G$. Recall that G is the group of all the $\mathbb{F}_p$ linear combinations of the α_i's. Concerning the Goppa polynomial, for all $\ell \in [0, \ldots, p^\lambda - 1]$, $g_\ell \in G$ and $g \in G \mapsto g + g_\ell$ is a bijection of G over itself. As a consequence:

$$\Gamma(z - g_\ell) = \gamma \left(\prod_{g \in G} (z - g_\ell - g) \right) = \gamma \left(\prod_{g \in G} (z - g) \right) = \Gamma(z).$$

This proves that Proposition 2 applies for all σ_ℓ with $0 \leqslant \ell \leqslant p^\lambda - 1$. Hence $\mathscr{G}(\boldsymbol{x}, \Gamma(z))$ admits an automorphism group of cardinality at least p^λ.
Finally, the previous results allow to prove Equation (7), linking the multipliers of an alternant representation of $\mathscr{G}(\boldsymbol{x}, \Gamma(z))$. Indeed, the relation between the support and multiplier vector is given for a Goppa code by $y_i = \Gamma(x_i)^{-1}$. By combining relations (4) and (5), for $\ell, 0 \leqslant \ell \leqslant p^\lambda - 1$, it holds that:

$$y_{it+\ell} = \Gamma(x_{it+\ell})^{-1} = \Gamma(x_{it} + g_\ell)^{-1} = \Gamma(x_{it})^{-1} = y_{it}.$$

□

[35][Algorithm 1, l. 19] and [3, Algorithm 3] generate the same supports as in Theorem 2, but consider only polynomials $\gamma(z)$ of degree $t_0 = 1$. Thus, we can obtain from Theorem 2 more codes than with the constructions provided in [35,3]. We conclude this part by a simple corollary of Theorem 2, generalizing to any characteristic a fact proven in char. 2 by [22].

Corollary 1. *Let $\mathscr{C} = \mathscr{G}(\boldsymbol{x}, \Gamma(z))$ be a Goppa code constructed as in Theorem 2. Let $\mathcal{A}_{t_0 p^\lambda}(\boldsymbol{x}, \boldsymbol{y})$ be an alternant description of $\mathscr{C}$. Then, the multiplier vector $\boldsymbol{y}$ satisfies the simple relation:*

$$y_i = \boldsymbol{y}_{\lfloor i/p^\lambda \rfloor p^\lambda}, \text{ for all } i,\ 0 \leqslant i \leqslant n-1. \tag{7}$$

3.2 Folding Symmetric Codes

Theorem 2 allows to construct a compact Goppa code from a regular Goppa code. In this part, we show that we can somewhat reverse this process. That is, we can retrieve a generator matrix of a code with the same set of multipliers as the original code used to construct the compact code. This can be done thanks to the so-called *folding* procedure.

Definition 5 (Folded code). *Let $\mathscr{C}$ be a code of length n with non-trivial automorphism σ of order ℓ, such that all the orbits have same size ℓ. We can split the code positions in $s = n/\ell$ orbits $\{i, \sigma(i), \sigma^2(i), \ldots, \sigma^{\ell-1}(i)\}$ and choose one representative $i_0, i_1, \ldots, i_{s-1}$ for each orbit (for instance the smallest one). The* folded code *of $\mathscr{C}$ with respect to σ, denoted by $\overline{\mathscr{C}^\sigma}$, is a code of length s which is given by the set of words $\overline{\boldsymbol{c}^\sigma} \stackrel{def}{=} \left(\sum_{u=0}^{\ell-1} c_{\sigma^u(i_j)}\right)_{0 \leqslant j \leqslant s-1}$, where $\boldsymbol{c}$ ranges over $\mathscr{C}$.*

Building a generator matrix of the folded code simply consists in summing the coordinates of each row of a generator matrix of $\mathscr{C}$ over the orbits of σ. This can be done efficiently. We now prove that folding a QM Goppa codes yields a QM Goppa codes but with a smaller automorphism group.

Theorem 3. *Let $\mathrm{char}(\mathbb{F}_{q^m}) = p$. Let $\mathscr{C}$ be a Goppa code $\mathscr{G}(\boldsymbol{x}, \Gamma(z))$ with length $n_0 p^\lambda$ and order $t_0 p^\lambda$ as constructed in Theorem 2. There exist $\alpha_0, \ldots, \alpha_{\lambda-1} \in \mathbb{F}_{q^m}$ and $\gamma \in \mathbb{F}_{q^m}[z]$ satisfying conditions (4) and (5). Let σ_1 be the code permutation defined by $\sigma_1(i) = i \ominus 1$ for all $i, 0 \leqslant i \leqslant n-1$. Then, there exist $\boldsymbol{x}' \in \mathbb{F}_{q^m}^{n/p}$ and a polynomial $\Gamma_1 \in \mathbb{F}_{q^m}[z]$ of degree $\deg(\Gamma)/p$ satisfying:*

$$x'_j = x_{jp}^p - \alpha_0^{p-1} x_{jp}, \text{ for all } j, 0 \leqslant j \leqslant n/p - 1, \tag{8}$$

$$\Gamma_1\left(z^p - \alpha_0^{p-1} z\right) = \Gamma(z), \tag{9}$$

such that $\overline{\mathscr{C}^{\sigma_1}} \subseteq \mathscr{G}(\boldsymbol{x}', \Gamma_1)$. Moreover, $\mathscr{G}(\boldsymbol{x}', \Gamma_1)$ has an automorphism group of size at least $p^{\lambda-1}$. Let $\mathcal{A}_{t_0 p^\lambda}(\boldsymbol{x}, \boldsymbol{y})$ (resp. $\mathcal{A}_{t_0 p^{\lambda-1}}(\boldsymbol{x}', \boldsymbol{y}')$) be the alternant description of $\mathscr{G}(\boldsymbol{x}, \Gamma(z))$ (resp. $\mathscr{G}(\boldsymbol{x}', \Gamma_1(z))$). Then, the multiplier vector $\boldsymbol{y}'$ is given by:

$$y'_i = y_{ip}, \text{ for all } i, 0 \leqslant i \leqslant n/p - 1. \tag{10}$$

Proof. As in Theorem 2, we denote by G the group of order p^λ generated by the α_i's. We define $\phi : z \in \mathbb{F}_{q^m} \mapsto z^p - \alpha_0{}^{p-1} z \in \mathbb{F}_{q^m}$. Remark that the map ϕ is additive: $\phi(z+z') = \phi(z) + \phi(z')$ and that $\phi(\alpha_0) = 0$. This implies that ϕ is constant over each coset of the subgroup $\langle \alpha_0 \rangle = \{\alpha_0, \ldots, (p-1)\alpha_0\}$ since for all $\ell \in \mathbb{F}_p$, $\phi(z + \ell\alpha_0) = \phi(z)$. Pick G_0 a set of representatives of the cosets of $\langle \alpha_0 \rangle$, a classical group theorem ensures that $G_0 \simeq_\phi G^*$, where $G^* = \phi(G)$ has $p^{\lambda-1}$ elements. We will also use the following facts:

$$\prod_{\ell=0}^{p-1} (X - \ell\alpha_0) = \phi(X) \quad \text{and} \quad \sum_{\ell=0}^{p-1} \frac{1}{X - \ell\alpha_0} = \frac{-\alpha_0^{p-1}}{\phi(X)}.$$

The first follows by observing that $\phi(\ell\alpha_0) = 0$ and by degree considerations and the second by observing that $\sum_{\ell=0}^{p-1} \frac{1}{X-\ell\alpha_0} = \frac{\phi'(X)}{\phi(X)} = \frac{-\alpha_0^{p-1}}{\phi(X)}$. We write $\prod_{g\in G}(z-g) =$

$$\prod_{g_0\in G_0}\left(\prod_{\ell=0}^{p-1}(z-g_0-\ell\alpha_0)\right) = \prod_{g_0\in G_0}\phi(z-g_0) = \prod_{g_0\in G_0}\big(\phi(z)-\phi(g_0)\big) = \prod_{g^*\in G^*}\big(\phi(z)-g^*\big).$$

From Theorem 2, we have $\Gamma(z) = \gamma\left(\prod_{g\in G}(z-g)\right)$. We define $\Gamma_1(z) \stackrel{\text{def}}{=} \gamma\big(\prod_{g^*\in G^*}(z-g^*)\big)$. This gives Γ_1 of degree $\deg(\Gamma)/p$ and such that $\Gamma_1(\phi(z)) = \Gamma(z)$. This proves (9). For (7), we can see that $y'_i = \Gamma_1(x'_i)^{-1} = \Gamma_1\big(\phi(x_{ip})\big)^{-1} = \Gamma(x_{ip})^{-1} = y_{ip}$.
Now, we show the inclusion $\overline{\mathscr{C}^{\sigma_1}} \subseteq \mathscr{G}(\boldsymbol{x}', \Gamma_1)$. Let $\boldsymbol{c}' = \overline{\boldsymbol{c}^{\sigma_1}} \in \overline{\mathscr{C}^{\sigma_1}}$, with $\mathbf{c} \in \mathscr{C}$. Thanks to Condition (4) of Theorem 2, we have for all $\ell, 0 \leqslant \ell < p$, $\phi(x_{jp\oplus\ell}) = \phi(x_{jp} + \ell\alpha_0) = \phi(x_{jp})$. Thus:

$$\begin{aligned} P_{\boldsymbol{c}',\boldsymbol{x}'} = \sum_{j=0}^{n/p-1}\frac{c'_j}{z-x'_j} &= \sum_{j=0}^{n/p-1}\frac{\sum_{u=0}^{p-1} c_{\sigma_1^u(i_j)}}{z-\phi(x_{jp})} = \sum_{j=0}^{n/p-1}\frac{c_{jp}+\cdots+c_{jp\oplus p-1}}{z-\phi(x_{jp})} \\ &= \sum_{j=0}^{n/p-1}\frac{c_{jp}}{z-\phi(x_{jp})} + \cdots + \frac{c_{jp\oplus(p-1)}}{z-\phi(x_{jp\oplus(p-1)})} = \sum_{i=0}^{n-1}\frac{c_i}{z-\phi(x_i)} = P_{\boldsymbol{c},\phi(\boldsymbol{x})}. \end{aligned}$$

According to Proposition 1, all the codewords $\boldsymbol{c}, \boldsymbol{c}^{\sigma_1}, \ldots, \boldsymbol{c}^{\sigma_{p-1}}$ belong to $\mathscr{G}(\boldsymbol{x}, \Gamma(z))$ Hence, we have that $P_{\boldsymbol{c},\boldsymbol{x}} + \cdots + P_{\boldsymbol{c}^{\sigma_{p-1}},\boldsymbol{x}} \equiv 0 \bmod \Gamma_1\big(\phi(z)\big) (= \Gamma(z))$. By a change of indices $i \mapsto i \ominus \ell$ in each $P_{\boldsymbol{c}^{\sigma_\ell},\boldsymbol{x}}$, we get:

$$P_{\boldsymbol{c},\boldsymbol{x}} + \cdots + P_{\boldsymbol{c}^{\sigma_{p-1}},\boldsymbol{x}} = \sum_{i=0}^{n-1}\sum_{\ell=0}^{p-1}\frac{c_i}{z-x_{i\oplus\ell}} = \sum_{i=0}^{n-1}\left(\sum_{\ell=0}^{p-1}\frac{c_i}{(z-x_i)-\ell\alpha_0}\right) = \sum_{i=0}^{n-1}\frac{(-\alpha_0^{p-1})c_i}{\phi(z)-\phi(x_i)} = (-\alpha_0^{p-1})P_{\boldsymbol{c}',\boldsymbol{x}'}\big(\phi(z)\big).$$

We can write that $P_{\boldsymbol{c}',\boldsymbol{x}'}(\phi(z)) = C(z)\Gamma_1(\phi(z))$, where $C(z)$ is a rational fraction. Remark that C is invariant by $z \mapsto z - \alpha_0$. It is then a classical result in invariant theory to write $C(z)$ under the form $C(z) = D(\phi(z))$. We conclude that $P_{\boldsymbol{c}',\boldsymbol{x}'}(z) = D(z)\Gamma_1(z)$, so that $\overline{\mathscr{C}^{\sigma_1}} \subseteq \mathscr{G}(\boldsymbol{x}', \Gamma_1)$.
Now, we prove that $\mathscr{G}(\boldsymbol{x}', \Gamma_1)$ is a code with automorphism group of cardinality $p^{\lambda-1}$. The idea is to show that $\boldsymbol{x}'$ and Γ_1 satisfy Conditions (4) and (5) of Theorem 2. We recall that the support $\boldsymbol{x}'$ is defined by $x'_i = x_{ip}^p - \alpha_0^{p-1}x_{ip} = \phi(x_{ip})$. Also, $\boldsymbol{x}$ satisfies (4), i.e. $x_i = \widetilde{x}_{\lfloor i/p^\lambda \rfloor} + \sum_{j=0}^{\lambda-1} i_j\alpha_j$ when $i \equiv \sum_{j=0}^{\lambda-1} i_j p^j \bmod p^\lambda$. Now pick i in $[0, \ldots, n/p-1]$. Clearly, we have $0 \leqslant ip \leqslant n-1$, and the decomposition of $ip \bmod p^\lambda$ is the decomposition of $i \bmod p^\lambda$ shifted by one position $(0, i_0, \ldots, i_{\lambda-2})$. Thus, by (4) we get $x_{ip} = \widetilde{x}_{\lfloor (ip)/p^\lambda \rfloor} + \sum_{j=1}^{\lambda-1} i_{j-1}\alpha_j$. Now, we apply the additive map ϕ, and use the fact that, as the i_j's are elements of $\{0, \ldots, p-1\}$, they satisfy $i_j^p = i_j$, so that $\phi(i_j\alpha_j) = i_j\phi(\alpha_j)$. Consequently:

$$\begin{aligned} x'_i &= \phi(\widetilde{x}_{\lfloor (ip)/p^\lambda \rfloor}) + \textstyle\sum_{j=1}^{\lambda-1}\phi(i_{j-1}\alpha_j), \\ &= \widetilde{x'}_{\lfloor (i)/p^{\lambda-1} \rfloor}) + \textstyle\sum_{j=0}^{\lambda-2} i_j\alpha'_j. \end{aligned}$$

We naturally set $\widetilde{x'}_i = \phi(\widetilde{x}_i)$ for $i \in [0, \ldots, n_0-1]$ and $\alpha'_i = \phi(\alpha_{i+1})$ for $i \in [0, \ldots, \lambda-2]$, and we see that $\boldsymbol{x}'$ satisfies (4) with a sequence of length $\lambda - 1$. Finally, $\Gamma_1(z)$ is the associated Goppa polynomial since G^* is obviously generated by the α'_i which are $\mathbb{F}_p-$independent (as a vanishing linear combination of the $\phi(\alpha_{i+1})$'s would give immediately a vanishing combination of the α_i's). According to Theorem 2, $\mathscr{G}(\boldsymbol{x}', \Gamma_1)$ has an automorphism group of size $p^{\lambda-1}$. □

A consequence of this theorem is that by folding successively with respect to all the automorphisms of the code, we obtain a sequence of Goppa codes whose parameters are smaller and smaller, and all sharing the same set of multipliers.

Remark 2. The dimension of the folded code $\overline{\mathscr{C}^{\sigma_1}}$ is a point of uncertainty. However, the alternant codes recommended for cryptographic use have all public matrices in systematic form $\big(\boldsymbol{A}\ \boldsymbol{I}_k\big)$ (for a code of dimension k). Then, it is clear that the matrix generating $\overline{\mathscr{C}^{\sigma_1}}$ contains a $k/p \times k/p$ identity block, so we know that $\overline{\mathscr{C}^{\sigma_1}}$ has parameters $[n/p, k/p, t/p]$.

4 Improved Algebraic Cryptanalysis

4.1 Algebraic Key-Recovery Attack on Alternant Codes

We start by explaining how [22] derived the algebraic system (1). Recall that the public code in McEliece is permutation-equivalent [40,30] to a secret q-ary alternant code. Thus, the public code is still a q-ary alternant code $\mathscr{A}_t(\boldsymbol{x}, \boldsymbol{y})$ defined by the public generator matrix $\boldsymbol{G} = (g_{i,j}) \in \mathbb{F}_q^{k\times n}$. The couple of vectors $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{F}_{q^m}^n \times \mathbb{F}_{q^m}^n$ permits to decode efficiently the public code, and completely break the cryptosystem. Stated differently, $(\boldsymbol{x}, \boldsymbol{y})$ is a key which is equivalent to the secret-key.
Let $\boldsymbol{X} \stackrel{\text{def}}{=} (X_0, \ldots, X_{n-1})$ and $\boldsymbol{Y} \stackrel{\text{def}}{=} (Y_0, \ldots, Y_{n-1})$ be two sets of variables corresponding to the unknown support $\boldsymbol{x}$ and multiplier $\boldsymbol{y}$ respectively. [22] used the fact that $\boldsymbol{V}_t(\boldsymbol{X}, \boldsymbol{Y})$ is a parity-check matrix (Definition 1) of the public-code. This means that $\boldsymbol{V}_t(\boldsymbol{X}, \boldsymbol{Y})\boldsymbol{G}^T = \boldsymbol{0}$ holds and yields:

$$\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}} \stackrel{\text{def}}{=} \bigcup_{\ell=0}^{t-1} \left\{ \sum_{j=0}^{n-1} g_{i,j} Y_j X_j^\ell = 0 \quad | \quad 0 \leqslant i \leqslant k-1 \right\}. \tag{11}$$

The equations occurring in this system have a particular structure.

Definition 6. *Let $f \in \mathbb{F}[\boldsymbol{X}, \boldsymbol{Y}]$. We shall say that f is* **bi-homogeneous** *of* **bi-degree** (d_1, d_2) *if:*

$$f(\alpha\, \boldsymbol{X}, \beta\, \boldsymbol{Y}) = \alpha^{d_1}\beta^{d_2} f(\boldsymbol{X}, \boldsymbol{Y}),\ \forall (\alpha, \beta) \in \mathbb{F} \times \mathbb{F}.$$

Thus, f is **bi-linear** *if is is of* **bi-degree** $(1, 1)$*. Also, if $p = \mathrm{char}(\mathbb{F})$, then we shall say that f is* **quasi bilinear** *[23] if it is of bi-degree (p^u, p^v) for $u, v \geqslant 0$*

It is clear that $(\boldsymbol{x}, \boldsymbol{y})$ is a solution of this system. Observe also that $\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}}$ is very structured: the only monomials occurring are of the form $Y_j X_j^\ell$ with $0 \leqslant \ell \leqslant t-1$. Furthermore, $\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}}$ becomes linear as soon as the variables of the block $\boldsymbol{X}$ are fixed. We hence obtain kt linear equations with n unknowns $\boldsymbol{Y}$. The equations obtained by fixing the variables of $\boldsymbol{Y}$ in $\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}}$ are not directly linear. They are of the form $\sum_{j=0}^{n-1} g'_{i,j} X_j^\ell = 0$ with $0 \leqslant i \leqslant n-1$. However, we can extract a linear system by only considering the equations with exponents $\ell = p^u$ provided that $0 \leqslant u \leqslant \lfloor \log_p(t-1) \rfloor$. We then apply the map $z \mapsto z^{q^m/p^u}$, which is additive in characteristic p, to this subset of equations. In other words, from the equation $\sum_{j=0}^{n-1} g'_{i,j} X_j{}^{p^u} = 0$ we deduce that $\sum_{j=0}^{n-1} \left(g'_{i,j}\right)^{q^m/p^u} X_j^{q^m} = 0$, which yields the linear equation $\sum_{j=0}^{n-1} \left(g'_{i,j}\right)^{q^m/p^u} X_j = 0$. This allows to obtain a linear system with $k\big(\lfloor \log_p(t-1) \rfloor + 1\big)$ equations.

Fact 4 *If the secret support $\boldsymbol{x}$ (resp. secret multiplier $\boldsymbol{y}$) is known then solving $\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}}$ reduces to solve a linear system having kt (resp. $k(\lfloor \log_p(t-1) \rfloor + 1)$) equations in n variables.*

As explained in [23], the k linear equations involving only the set of variables $\boldsymbol{Y}$ can be used to eliminate some variables in the block $\boldsymbol{Y}$. By assumption, the public-code defined by $\boldsymbol{G}$ is of dimension k. Up to Gaussian elimination (and possibly reordering the positions), one can always assume that $\boldsymbol{G}$ is in systematic form $\left(\boldsymbol{A}\ \boldsymbol{I}_k\right)$ where $\boldsymbol{I}_k$ is the $k \times k$ identity matrix and $\boldsymbol{A} = (a_{i,j}) \in \mathbb{F}_q^{k \times n-k}$. Thus, for all $\ell, 0 \leqslant \ell \leqslant t-1$:

$$Y_{n-k+i} X_{n-k+i}^{\ell} = - \sum_{j=0}^{n-k-1} a_{i,j} Y_j X_j^{\ell}, \quad \forall i, 0 \leqslant i \leqslant k-1. \tag{12}$$

By focusing on equations with $\ell = 0$, we construct a new polynomial system $\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}'}$ – where $\boldsymbol{Y}' \stackrel{\text{def}}{=} (Y_0, \ldots, Y_{n-k-1})$ – in which the variables $(Y_k, \ldots, Y_{n-1})$ have been eliminated:

$$\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}'} \stackrel{\text{def}}{=} \bigcup_{\ell=1}^{t-1} \left\{ \sum_{j=0}^{n-k-1} a_{i,j} Y_j \left(X_j^{\ell} - X_{n-k+i}^{\ell} \right) = 0 \ \mid \ 0 \leqslant i \leqslant k-1 \right\}. \tag{13}$$

4.2 Algebraic Description of Goppa Codes

We have summarized in Section 4.1 the algebraic approach described in [22,23]. In this part, we address a question reported by few authors [2,1]: how to describe completely a Goppa (resp. binary Goppa) decoder with the algebraic attack of [22]. Indeed, when the public code is a Goppa code, the equivalent key $(\boldsymbol{x}, \boldsymbol{y})$ vanishes $\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}'}$ but also extra algebraic equations. Precisely, Definition 2 implies that there exists $\Gamma(z)$ in $\mathbb{F}_{q^m}[z]$ of degree t such that $\sum_{\ell=0}^{t} \gamma_\ell Y_j X_j^{\ell} = Y_j \Gamma(X_j) = 1$. The coefficients $\gamma_0, \ldots, \gamma_t \in \mathbb{F}_q^m$ of $\Gamma(z)$ are unknown. However, thanks to (12), we can write equations where almost all of them have disappeared. For any $i, 0 \leqslant i \leqslant k-1$:

$$\sum_{\ell=0}^{t} \gamma_\ell \underbrace{\left(Y_{n-k+i} X_{n-k+i}^{\ell} + \sum_{j=0}^{n-k-1} a_{i,j} Y_j X_j^{\ell} \right)}_{=0 \text{ when } \ell<t} = \underbrace{\sum_{\ell=0}^{t} \gamma_\ell Y_{n-k+i} X_{n-k+i}^{\ell}}_{=Y_{n-k+i}\Gamma(X_{n-k+i})} + \sum_{j=0}^{n-k-1} a_{i,j} \underbrace{\sum_{\ell=0}^{t} \gamma_\ell Y_j X_j^{\ell}}_{=Y_j \Gamma(X_j)}. \tag{14}$$

This yields $\gamma_t \left(Y_{n-k+i} X_{n-k+i}^{t} + \sum_{j=0}^{n-k-1} a_{i,j} Y_j X_j^{t} \right) = 1 + \sum_{j=0}^{n-k-1} a_{i,j}$, $\forall i, 0 \leqslant i \leqslant k-1$. We can include such equations into $\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}'}$ by adding only one variable for γ_t^{-1}. After performing the same elimination as (13), we obtain a new polynomial system $\mathsf{G}_{\boldsymbol{X},\boldsymbol{Y}'}$ dedicated to q-ary Goppa codes:

$$\mathsf{G}_{\boldsymbol{X},\boldsymbol{Y}'} \stackrel{\text{def}}{=} \mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}'} \cup \left\{ \sum_{j=0}^{n-k-1} a_{i,j} Y_j \left(X_j^{t} - X_{n-k+i}^{t} \right) = \frac{1}{\gamma_t} \left(1 + \sum_{j=0}^{n-k-1} a_{i,j} \right) \mid 0 \leqslant i \leqslant k-1 \right\}. \tag{15}$$

Remark 3. The system (15) can also be deduced from a known property of Goppa codes. For instance [7, Proposition 2] shows that the *extended* code of a Goppa code has a parity-check matrix $\tilde{\boldsymbol{H}}$ with a very specific form involving $\boldsymbol{V}_{t+1}(\boldsymbol{x}, \boldsymbol{y})$ and γ_t^{-1}. The system $\boldsymbol{G}_{pub} \tilde{\boldsymbol{H}} = \boldsymbol{0}$ is exactly $\mathsf{G}_{\boldsymbol{X},\boldsymbol{Y}'}$.

Binary case ($q = 2$). The case of binary Goppa codes is even more specific. Such codes can be viewed as an alternant codes $\mathscr{A}_t(\boldsymbol{x}, \boldsymbol{y})$ with $y_j = \Gamma(x_j)^{-1}$ for all $j, 0 \leqslant j \leqslant n-1$ but also described

as a binary alternant codes $\mathscr{A}_{2t}(\boldsymbol{x}, \boldsymbol{y}^2)$ (Theorem 1). This brings new equations to the system $\mathsf{G}_{\boldsymbol{X},\boldsymbol{Y}'}$ which are defined by $Y_{n-k+i}^2 X_{n-k+i}^{\ell} = -\sum_{j=0}^{n-k-1} a_{i,j} Y_j^2 X_j^{\ell}$ where $0 \leqslant i \leqslant k-1$ and $0 \leqslant \ell \leqslant 2t-1$. This enables to define a specific algebraic system $\mathsf{McE}_{\boldsymbol{X},\boldsymbol{Y}'}$ dedicated to McEliece's cryptosystem:

$$\mathsf{McE}_{\boldsymbol{X},\boldsymbol{Y}'} \stackrel{\text{def}}{=} \mathsf{G}_{\boldsymbol{X},\boldsymbol{Y}'} \cup \bigcup_{\ell=1}^{2t-1} \left\{ \sum_{j=0}^{n-k-1} a_{i,j} Y_j^2 \left(X_j^{\ell} - X_{n-k+i}^{\ell} \right) = 0 \mid 0 \leqslant i \leqslant k-1 \right\}. \tag{16}$$

4.3 Structural Elimination

We present now a new method to eliminate some variables from the block $\boldsymbol{X}$ in our algebraic systems. We restrict our attention to $\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}'}$ (a similar analysis holds for $\mathsf{G}_{\boldsymbol{X},\boldsymbol{Y}'}$ and $\mathsf{McE}_{\boldsymbol{X},\boldsymbol{Y}'}$). The idea is to consider a suitable subset of the equations occurring; more precisely the equations of bi-degree $(p^u, 1)$ where $0 \leqslant u \leqslant \lfloor \log_p(t-1) \rfloor$. The only restriction is that the characteristic p has to be smaller than t. The technique is inspired from ideas proposed to construct a distinguisher for alternant and Goppa codes [21,20]. The principle is to suitably combine monomials $Y_{n-k+i} X_{n-k+i}^{p^u}$, with $0 \leqslant u \leqslant \lfloor \log_p(t-1) \rfloor$ and $0 \leqslant i \leqslant k-1$. Thanks to (12), the trivial identities $\left(Y_{n-k+i} X_{n-k+i}^{p^u} \right)^p = Y_{n-k+i}^{p-1} Y_{n-k+i} X_{n-k+i}^{p^{u+1}}$ can be rewritten with $0 \leqslant u \leqslant \lfloor \log_p(t-1) \rfloor - 1$ and $0 \leqslant i \leqslant k-1$ as:

$$\left(-\sum_{j=0}^{n-k-1} a_{i,j} Y_j X_j^{p^u} \right)^p = \left(-\sum_{j=0}^{n-k-1} a_{i,j} Y_j \right)^{p-1} \left(-\sum_{j=0}^{n-k-1} a_{i,j} Y_j X_j^{p^{u+1}} \right). \tag{17}$$

Equations defined in (17) only contains $X_0, \ldots, X_{n-k-1}$. We then set $\boldsymbol{X}' \stackrel{\text{def}}{=} (X_0, \ldots, X_{n-k-1})$. Thus, by cleaning-up relations (17), we define $\mathsf{elimA}_{\boldsymbol{X}',\boldsymbol{Y}'} \stackrel{\text{def}}{=}$:

$$\bigcup_{u=0}^{\lfloor \log_p(t-1) \rfloor - 1} \bigcup_{i=0}^{k-1} \left\{ \sum_{j=0}^{n-k-1} X_j^{p^{u+1}} \left(a_{i,j}^p Y_j^p - a_{i,j} Y_j \left(-\sum_{j'=0}^{n-k-1} a_{i,j'} Y_{j'} \right)^{p-1} \right) = 0 \right\}. \tag{18}$$

The polynomials of this system are quasi-bilinear [23] with bi-degree (p^{u+1}, p). Thus, we replaced equations of bi-degree $(p^u, 1)$ by ones of higher bi-degree (p^{u+1}, p). On the other hand, we reduced the number of variables from $2n-k$ to $2(n-k)$. In Section 5, we can see that the structural elimination allows to get significant practical improvements despite the degree increase.
We can also perform the elimination over the additive equations in $\mathsf{G}_{\boldsymbol{X},\boldsymbol{Y}'}$ and $\mathsf{McE}_{\boldsymbol{X},\boldsymbol{Y}'}$. To eliminate $X_{n-k}, \ldots, X_{n-1}$ in (16), we develop $Y_{n-k+i}^2 X_{n-k+i}^{p^u} = Y_{n-k+i} Y_{n-k+i} (X_{n-k+i})^{p^u}$ with Equation (16). We can perform the elimination on Equation (15) when the Goppa polynomial has degree $t = p^{\lambda}$. To do so, develop $\left(Y_{n-k+i} X_{n-k+i}^{p^{(\lambda-1)}} \right)^p = Y_{n-k+i}^{p-1} \left(Y_{n-k+i} X_{n-k+i} \right)^t$ thanks to (14). Table 1 summarizes the number of equations, number of variables in each block, and the structure of the different algebraic systems introduced in this part. According to [22, Proposition 1], we can fix in $\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}'}$ one of the Y_i''s and two X_i's to arbitrary values. This is due to the fact that an alternant code admits several equivalent representations. In the systems $\mathsf{G}_{\boldsymbol{X},\boldsymbol{Y}'}$ and $\mathsf{McE}_{\boldsymbol{X},\boldsymbol{Y}'}$, we choose to fix the leading coefficient of the Goppa polynomial instead of one Y_i. Remark also that some variables have to be added to be sure that the variety associated to $\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}'}$ has few solutions. In

particular, we have to remove parasite solutions corresponding to $X_i = X_j$ and $Y_j = 0$. A classical way to do so that is to introduce new variables u_{ij} and v_i and add to $\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}'}$ equations of the form $u_{ij} \cdot (X_i - X_j) + 1 = 0$ and $v_i \cdot Y_i + 1 = 0$. In practice, to avoid adding too many new variables, we have added only few of them namely 4 or 5. In order to ease the explanation, we ignored this point on paper. However, we emphasize that we are always adding such relations to perform the experiments of Section 5.

	$\#\boldsymbol{X}$	$\#\boldsymbol{Y}'$	Equations
$\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}'}$	$n-2$	$n-k-1$	$k \cdot (t-1)$ eq. of bi-degree $(u,1)$, with $1 \leqslant u \leqslant t-1$
$\mathsf{G}_{\boldsymbol{X},\boldsymbol{Y}'}$	$n-2$	$n-k$	$\#\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}'}$ eq. $+ k$ eq. of bi-degree $(t,1)$
$\mathsf{McE}_{\boldsymbol{X},\boldsymbol{Y}'}$	$n-2$	$n-k$	$\#\mathsf{G}_{\boldsymbol{X},\boldsymbol{Y}'}$ eq . $+ k \cdot (2t-1)$ eq. of bi-degree $(u,2)$ with $1 \leqslant u \leqslant 2t-1$
$\mathsf{elimA}_{\boldsymbol{X}',\boldsymbol{Y}'}$	$n-k-2$	$n-k-1$	$k \cdot \lfloor \log_p(t-1) \rfloor$ eq. of bi-degree (p^u, p), with $1 \leqslant u \leqslant \lfloor \log_p(t-1) \rfloor$
$\mathsf{elimG}_{\boldsymbol{X}',\boldsymbol{Y}'}$	$n-k-2$	$n-k$	$\#\mathsf{elimA}_{\boldsymbol{X}',\boldsymbol{Y}'}$ eq. $+ k$ eq. of bi-degree $(t,1)$ with $t = p^\lambda$
$\mathsf{elimMcE}_{\boldsymbol{X}',\boldsymbol{Y}'}$	$n-k-2$	$n-k$	k eq. of bi-degree $(t,1)$ with $t = 2^\lambda + k \cdot \lfloor \log_2(2t-1) \rfloor$ eq. of bi-degree $(2^u, 2)$, with $0 \leqslant u \leqslant \lfloor \log_2(2t-1) \rfloor$

Table 1. Algebraic key-recovery systems for McEliece-like cryptosystems. The notation # denotes the size of the set considered. For $\mathsf{elimG}_{\boldsymbol{X}',\boldsymbol{Y}'}$ and $\mathsf{elimMcE}_{\boldsymbol{X}',\boldsymbol{Y}'}$, we assume that $t = p^\lambda$.

5 Practical Cryptanalysis of QM Schemes

From now on, as all the parameters proposed in the literature suggest [35,2,3], we consider only QM codes constructed using $\gamma(z) = z$ and $t_0 = 1$ (notations as in Theorem 2). Some specific codes proposed in [2,3] rely on a somewhat tweaked construction, which gives unusual t's ($t \neq p^\lambda$). This is not a limitation of our attack. We explain in 5.2 how to easily convert the case $t \neq p^\lambda$ into an equivalent problem where t is a pure power of p.

5.1 A New Strategy for Attacking QM Schemes

General Framework. We suppose that the public code was built with Theorem 2, so that $t = p^\lambda$ for $\lambda \geqslant 1$. The first part of the attack is to successively fold the public code. We obtain codes with same structure and same multipliers (Theorem 3). After s iterations ($0 \leqslant s \leqslant \lambda - 1$) of the folding process, the resulting code is of order $t_{(\lambda-s)} = t/p^{(\lambda-s)}$, length $n_s = n/p^{(\lambda-s)}$ and dimension $k_s = k/p^{(\lambda-s)}$. From Table 1, one can see that the corresponding algebraic systems will have then fewer variables and equations, same over-determination ratio, and equations of much smaller degrees. For Gröbner bases algorithms [17,18], this is a very good deal. For instance, compared to FOPT's experimental results [22,23], we obtained with this new strategy (Table 3) a speed-up wich can be as big as 45000 using on-the-shelf computer algebra system Magma [13] (whilst the results of [22,23] were obtained with the Fgb software[19]; an optimized C implementation of the F_5 algorithm [18]). This is the hardest part of our algebraic attack.
Once private support and multipliers are known for the folded code, we also know the multipliers of the public code (Theorem 3). Then, Fact 4 ensures that one can efficiently recover an equivalent decoder for the public code. We sum-up the strategy as follows (Steps 0) and 6) are exposed in 5.2):

0) (If t is not a pure power of p, expand the public QM code into a QM code whose order is a pure power of t. This expanded code has the same support as the public code (Theorem 5). We use then this expanded code as a public-key in the next steps.)
1) Fold iteratively the matrix of the public code to obtain a code with the same structure (alternant, or Goppa) but with smaller order of symmetry $\widetilde{t} < t$ (Theorem 3).
2) Construct the algebraic system $\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}'}$, $\mathsf{G}_{\boldsymbol{X},\boldsymbol{Y}'}$, or $\mathsf{McE}_{\boldsymbol{X},\boldsymbol{Y}'}$ (Section 4) from the folded code with or without structural elimination (Section 4.3).
3) Use Gröbner bases to recover the multiplier vector of the folded code (results in 5.3).
4) Expand the multiplier of the folded code to the multiplier of the public code (Theorem 3).
5) Solve the linear system (Fact 4) which allows to construct a decoder for the public code.
6) (If t is not a pure power of p, generate the system $\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}'}$ from the original (i.e. non expanded) public-key matrix, and solve the linear system obtained by plugging the support found at the previous step in $\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}'}$. This yields the multipliers of the initial public code, and then construct a decoder for the public-key).

Choice of the final $\widetilde{t}$ and of the algebraic system. Regardless of the solving algorithm, it is natural to consider folded code of order $\widetilde{t}$ as small as possible. However, there are lower bounds on $\widetilde{t}$. According to Table 1, $\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}'}$ is empty for a code of order 1, and $\mathsf{elimA}_{\boldsymbol{X}',\boldsymbol{Y}'}$ is empty for a code of order p (i.e. $\lambda = 1$). Thus, we need to take $\widetilde{t}$ not smaller than p for $\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}'}$ or $\widetilde{t}$ not smaller than p^2 for $\mathsf{elimA}_{\boldsymbol{X}',\boldsymbol{Y}'}$. Thus, the final $\widetilde{t}$ is determined by the choice to perform or not the structural elimination. We reported for each of our experiments the chosen $\widetilde{t}$. This is the major choice that an attacker has to make. It is actually not always straightforward to predict a priori which system between $\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}'}$ and $\mathsf{elimA}_{\boldsymbol{X}',\boldsymbol{Y}'}$ is easier to solve. Equations in $\mathsf{elimA}_{\boldsymbol{X}',\boldsymbol{Y}'}$ are of higher degrees than in $\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}'}$ but contain fewer variables. The relevant parameters to take into account are the characteristic (as the structural elimination increases the bi-degree by a factor of p), and the length of the code (the amount of variables in $\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}'}$ decreases when the length grows). For various parameters (notably for encryption parameters), we tried both and selected the most efficient.

5.2 Expanding a QM Code

Most of the results presented until now assume that $t = p^\lambda$. Whilst this a very natural requirement, it turns that some parameters proposed for signature in [2,3] use "degenerated" QM codes, where t is not a pure power of p. However, a parity-check matrix $\mathbf{h}$ of the public code still satisfies $h_{i,j} = h_{0,j\ominus i}$ for $0 \leqslant i \leqslant t-1$. Then, it turns out that the automorphism group is not of cardinality t but $\gcd(t, p^\lambda)$ where λ is such that $p^{\lambda-1} < t < p^\lambda$. As a consequence, we cannot fold the public code as many times as previously. To overcome this technical problem, we can however exploit the specific features of the QM codes used in [2,3]. We can retrieve an alternant code with the same support as the public code, but with a Goppa polynomial of degree p^λ, where we define $\lambda > 0$ as the smallest integer such that $t < p^\lambda$. We shall call $\mathscr{C}^{(t)}_{pub}$ the public code, and $\mathscr{C}^{(p^\lambda)}$ the code that we are trying to recover.

Theorem 5. *Let λ be the smallest integer such as $t < p^\lambda$. Let $\mathscr{C}^{(p^\lambda)}$ be the largest* QM *code stable by $\sigma_1, \ldots, \sigma_{p^\lambda - 1}$ (as defined in Proposition 1), and contained in $\mathscr{C}^{(t)}_{pub}$. Let $\boldsymbol{H} = (h_{i,j})$ be a parity-check matrix of $\mathscr{C}^{(t)}_{pub}$. We define a matrix $\Delta_{p^\lambda}(\boldsymbol{H}) = (h'_{i',j'})$ such that :*

$$h'_{ip^\lambda+\ell,j} = h_{i,j\ominus\ell}, \forall j, 0 \leqslant j < n, \ell, 0 \leqslant \ell < p^\lambda.$$

Then, $\mathscr{C}^{(p^\lambda)}$ is an alternant code with the same support as $\mathscr{C}^{(t)}_{pub}$ and parity-check matrix $\Delta_{p^\lambda}(\boldsymbol{H})$.

Thanks to this theorem (proven in Appendix A), we can explain the general strategy previously presented. Using Δ_{p^λ}, we can write a new public-matrix for $\mathscr{C}^{(p^\lambda)}$ (Step 0)). Then, we can recover the multiplier vector and the support of $\mathscr{C}^{(p^\lambda)}$ (Steps 1-2-3-4-5)). Since both codes share the same support, Fact 4 allows to recover the multipliers of $\mathscr{C}^{(t)}_{pub}$ by solving a linear system (Step 6)).

5.3 Experiments and Practical Results

We considered the parameters quoted in [2,35,3]. Remark that the public codes considered in these papers are alternant codes for $q > 2$, but (binary) Goppa codes for $q = 2$. So, we use in our experiments the systems $\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}'}, \mathsf{McE}_{\boldsymbol{X},\boldsymbol{Y}'}$ and their reduced versions. We also generated new parameters to see how our attack scales. To compute the security levels of these new parameters with respect to ISD [12,10,11,39], we used the ISDFQ software of C. Peters.[1]

Simplifying the Algebraic Systems for QM *Codes.* As we explained in 5.1, the final folded codes have order $\widetilde{t}$ equal to p or p^2. The algebraic system $\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}'}$ generated from the folded code can still be simplified by using Theorem 2. The folded code has a smaller but non-trivial automorphism groups. Consequently, we can use the linear relations (4) and (7) to decrease the number of variables. From now on, we will always assume that such linear relations are used in our systems.

Signature Schemes – Experimental Results. The QM codes used in signature schemes always have the maximal possible lengths, namely $n = q^m - t$. This leads to codes of relatively big lengths. The total numbers of variables in the systems $\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}'}$ can be also huge, typically up to $\geqslant 500$. However, $\mathsf{elimA}_{\boldsymbol{X}',\boldsymbol{Y}'}$ contains very few variables compared to the number of equations. For example, $\mathsf{elimA}_{\boldsymbol{X}',\boldsymbol{Y}'}$ contains 19671 equations and 21 variables for $(q = 3, m = 11)$. To compare both approaches on a tiny example, we picked a code with parameters $(q = 16, m = 3)$, length $n = 360$ and block size $t = 8$. Solving $\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}'}$ (containing 126 equations in 49 variables) took 311 seconds, whereas $\mathsf{elimA}_{\boldsymbol{X}',\boldsymbol{Y}'}$ contains 84 equations in only 8 variables and was solved in 0.01 second. Thus, we choose to perform the structural elimination in all possible cases. This permitted to mount a key-recovery for almost all the signature parameters proposed in [2,3]. In Table 2, we detail our experimental results. We have used MAGMA [13] (V2.17-1) to implement our attack. All the timings have been obtained on a 2.93 GHz Intel®. The Gröbner bases computation in MAGMA are performed with an optimized version of F_4 [17]. We have been able to break all the parameters proposed in [2,3] For most parameters proposed in [3], we have been able to recover the secret in few seconds (the last row took less than 2 hours). As a conclusion, the use of QM codes introduces a fatal weakness in the signature context. In view of our new attack, it seems impossible to find secure parameters for QM (resp. QD) signature schemes.
For a public alternant code with $t = p$, neither the folding nor the structural elimination are possible (as explained in 4.3). For those cases, we pick equations from the first block of rows. That is:

$$\sum_{j=0}^{n-k-1} a_{i,j} Y_j \left(X_j^\ell - X_{n-k+i}^\ell \right) = 0,$$

[1] `http://christianepeters.wordpress.com/publications/tools/`

Solving $\mathsf{elimMcE}_{\boldsymbol{X}',\boldsymbol{Y}'}$.

$p=2$	m	n	t	$\widetilde{t}=p$	n_0	$(2m-2)$ unk.	$2(n_0-m)$ equ.	Bi-degree	Magma	Sec. level
2	13	8176	16	2	511	25	996	$(2,1),(2,2)$	1.9 s.	84
2	13	8176	15	2	511	25	996		1.9 s.	81
2	14	16368	14	2	1023	27	2018		3.0 s.	84
2	14	16368	13	2	1023	27	2018		3.0 s.	81
2	15	32752	12	2	2047	29	4064		5.4 s.	82

$\mathsf{elimA}_{\boldsymbol{X}',\boldsymbol{Y}'}$ solving timings

$p>2$	m	n	t	$\widetilde{t}=p^2$	n_0	$(2m-1)$ unk.	(n_0-m) equ.	Bi-degree (p,p)	Magma	
3	11	177048	9	9	19682	21	19671	$(3,3)$	3.4 s.	80
5	8	390495	15	25	15624	15	15616	$(5,5)$	82 s.	128

Solving $\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}'}$ when $t=p$ (so, without folding).

$p>2$	m	n	t	$\widetilde{t}=t$	n_0	# unk.	$\leqslant (p-1)t$ equ.	Bi-degree	Magma	
13	4	28509	13	13	2196	12	8	$(1,i), 1\leqslant i\leqslant 7$	0.05 s	80
13	5	371228	13	13	28560	12	10	$(1,i), 1\leqslant i\leqslant 7$	5873 s.	112

Table 2. Practical attacks against signature schemes with parameters from [2,3].

with $i \in \{0,\ldots,t-1\}$ (instead of $0 \leqslant i \leqslant k-1$). This system only involves X_0, X_1, and $X_t, X_{2t}, \ldots, X_{(n_0-m)t}$, contains up to $t(p-1)$ equations and $2m-1$ variables. The strategy was very successful for small m's ($m \leqslant 6$).

Encryption – Experimental Results. The algebraic systems are harder to solve for encryption parameters. In addition of Magma, we also use Fgb to compute the Gröbner bases. For Fgb, we reported the basic number of expected operations. To estimate this number, we have mixed exhaustive search and Gröbner bases. This explains that we reported number of operations $\geqslant 2^{80}$. To see how the attack behaves, we also generated ourselves cryptographic secure parameters with isdfq (marked with a $\star$ in the table below). Our practical results are somewhat orthogonal with ISD. In most cases, the complexity of ISD grows whilst the efficiency of our attack increases. This is due to the fact that the folding process makes the value of t irrelevant for our attack; which is not the case for ISD. For encryption parameters, the choice between $\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}'}$ and $\mathsf{elimA}_{\boldsymbol{X}',\boldsymbol{Y}'}$ is less clear than in the signature context. Both systems have a reduced number of equations compared to the signature case. Thus, the higher degree induced by the structural elimination may be an obstacle to the resolution of $\mathsf{elimA}_{\boldsymbol{X}',\boldsymbol{Y}'}$. Solving this system proved to be very efficient for codes over $\mathbb{F}_q$ with $q=2^s$ and $s>1$, as we found multipliers and support up to 45000 times faster than FOPT's attack in [22], but is less efficient for $p \geqslant 3$. We mention that some parameters proposed could not be solved in practice. It is not unlikely that better results could be obtained in the future by using a specific Gröbner strategy for solving the systems (which are still structured). In any case, we see the experimental results as a practical validation of the structural weakness of compact codes coming from the folding process.

6 Concluding remarks

For a long time, the most dangerous attack against the McEliece cryptosystem based on Goppa codes had been message recovery attacks using generic decoding algorithms for linear codes. The

Solving $\mathsf{elimMcE}_{\boldsymbol{X}',\boldsymbol{Y}'}$ with $\widetilde{t}=p$.

$q=2$	m	n	t	$\widetilde{t}=2$	n_0	$2m-1$ unk.	$2(n_0-m)$ equ.	Magma	F_5/FGb	FOPT	Sec. level
2	16	4864	2^5	2	152	31	272	18 s.		N.A.	128
2	12	3200	2^7	2	25	23	22		$\leqslant 2^{83.5}$ op.	N.A.	128
2	14	5376	2^7	2	42	27	36		$\leqslant 2^{96.1}$ op.	N.A.	226
2	15	11264	2^9	2	22	29	14		$\leqslant 2^{146}$ op.	N.A.	256
2	16	6912	2^8	2	27	31	33		$\leqslant 2^{168}$ op.	N.A.	218
2	16	8192	2^8	2	32	31	32		$\leqslant 2^{157}$ op.	N.A.	256

Solving $\mathsf{elimA}_{\boldsymbol{X}',\boldsymbol{Y}'}$ with $\widetilde{t}=p^2$.

$q>2$	m	n	t	$\widetilde{t}=p^2$	n_0	$2m-1$ unk.	(n_0-m) equ.	Magma	F_5/FGb	FOPT	Sec. level
2^4	4	2048	2^6	4	32	8	28	0.01 s.		0.50 s	128
2^4	4	4096	2^7	4	32	8	28	0.01 s.		7.1 s	128
2^2	8	3584	2^6	4	56	15	48	0.04 s.		1,776 s	128
3	8	3645	3^4	9	45	15	37		$\leqslant 2^{44.5}$ op.	N.A.	224 $\star$
3	11	4860	3^4	9	60	21	49		3 d. 17h.	N.A.	192 $\star$
3	11	6885	3^4	9	85	21	64		181 s.	N.A.	261 $\star$
5	8	2500	5^2	5^2	100	15	92		2.9s	N.A.	107 $\star$
5	8	1375	5^2	5^2	55	15	47		160s	N.A.	85 $\star$
5	9	1750	5^2	5^2	70	17	61		728.4s	N.A.	89.8 $\star$
5	10	2000	5^2	5^2	80	19	70		5941.9s	N.A.	91.3 $\star$

Solving $\mathsf{A}_{\boldsymbol{X},\boldsymbol{Y}'}$ with $t=p$.

q	m	n	t	$\widetilde{t}=t$	n_0	# unk.	$\leqslant t(p-1)$ equ.	Magma	FOPT	Sec. level
167	3	668	167	167	4	5	7	0.020 s.	N.A.	80
241	3	964	241	241	4	5	7	0.020 s.	N.A.	112
41	3	451	41	41	11	14	24	0.030 s.	N.A.	80
5	5	1000	5^3	5	8	11	12	140 s.	N.A.	80
11	5	1089	11^2	11	9	12	20	225 s.	N.A.	112
7	5	735	7^2	7	15	18	60	900 s.	N.A.	80
7	6	1813	7^2	7	37	41	186	60 s.	N.A.	128

Table 3. Practical attacks against the encryption parameters proposed in [35,3]. N.A. means that the parameters could not be addressed in FOPT.

most threatening key recovery attack was just exhaustive search of the Goppa polynomial through generic algorithms for finding the permutation linking two equivalent codes. The latter turns out to be much more complex than generic decoding algorithms for usual parameters of the McEliece cryptosystem. This was the state of the art for a long time and was one of the reasons which justified to try to overcome the main drawback of the McEliece cryptosystem that was its large public key size by (possibly) losing a little bit in key security by suggesting alternant codes or Goppa codes with additional symmetries [8,35]. This was soon followed by a sequence of papers showing that key recovery attacks can be devastating in this case [22,23,43]. This was obtained by introducing new algebraic attacks obtained from an algebraic modeling of the Goppa key and then using Gröbner basis techniques to solve them. The binary parameters in [35] were not broken by this approach and subsequently a slightly more general symmetric McEliece scheme was proposed [3](by replacing the QD symmetry by the more general QM symmetry) and signature schemes were also proposed [2]. It was advocated that both of them were immune to the aforementioned algebraic attack when the Goppa codes were defined over prime fields.

This paper demonstrates several points. First regardless of the attack which is used to attack the scheme, there is an inherent weakness arising from Goppa codes with QM or QD symmetries, because it is possible to derive from the public key a much smaller public key corresponding to the folding of the original QM or QD code, where the reduction factor of the code length is precisely the order of the QM or QD group used for reducing the key size. From a structural-only security point of view, the security only relies on the small folded code and not on the bigger compact public matrix, because the algebraic structure original Goppa code can be be recovered from the algebraic structure of the folded Goppa code. Of course, the whole approach may stay valid as long as key recovery attacks on the smaller code stay more complex than message recovery attacks on the original system.
This raises the issue of studying, improving and trying to assess precisely the power of the algebraic attacks derived from the approach pioneered in [22]. One of the first point to understand is whether or not this algebraic approach is able to attack Goppa codes defined over prime fields. A second contribution of this paper is to show that QD or QM McEliece cryptosystems based on Goppa codes defined over prime fields can actually be attacked by first folding the code and then using an improved version of the algebraic modeling of [22]. This shows that QD or QM Goppa codes based on prime fields should not be considered immune to this kind of algebraic approach. Obviously this kind of attack needs in the future a better understanding. Obtaining overdefined algebraic systems helps in breaking them as shown by the attack of the signature schemes, and obtaining an accurate estimation of the complexity of the time complexity of this new attack would be a desirable goal.

7 Acknowledgements

We would like to thank (some) of the referees of EC'14 for helpful comments on a preliminary version of this paper. The work of the first and third authors has been partly supported by the French ANR-11-BS02-0013 HPAC project.

References

1. Morgan Barbier. Key reduction of mceliece's cryptosystem using list decoding. *CoRR*, abs/1102.2566, 2011.
2. Paulo S. L. M. Barreto, Pierre-Louis Cayrel, Rafael Misoczki, and Robert Niebuhr. Quasi-dyadic CFS signatures. In Xuejia Lai, Moti Yung, and Dongdai Lin, editors, *Inscrypt*, volume 6584 of *Lecture Notes in Computer Science*, pages 336–349. Springer, 2010.
3. Paulo S. L. M. Barreto, Richard Lindner, and Rafael Misoczki. Monoidic codes in cryptography. In Bo-Yin Yang, editor, *PQCrypto*, volume 7071 of *Lecture Notes in Computer Science*, pages 179–199. Springer, 2011.
4. Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 520–536. Springer, 2012.
5. T. P. Berger. Cyclic alternant codes induced by an automorphism of a GRS code. In R. Mullin and G. Mullen, editors, *Finite fields: Theory, Applications and Algorithms*, volume 225, pages 143–154, Waterloo, Canada, 1999. AMS, Contemporary Mathematics.
6. T. P. Berger. Goppa and related codes invariant under a prescribed permutation. *IEEE Trans. Inform. Theory*, 46(7):2628, 2000.
7. T. P. Berger. On the cyclicity of Goppa codes, parity-check subcodes of Goppa codes and extended Goppa codes. *Finite Fields and Applications*, 6:255–281, 2000.
8. T. P. Berger, P.L. Cayrel, P. Gaborit, and A. Otmani. Reducing key length of the McEliece cryptosystem. In Bart Preneel, editor, *Progress in Cryptology - Second International Conference on Cryptology in Africa (AFRICACRYPT 2009)*, volume 5580 of *Lecture Notes in Computer Science*, pages 77–97, Gammarth, Tunisia, June 21-25 2009.

9. D. J. Bernstein, T. Lange, and C. Peters. Attacking and defending the McEliece cryptosystem. In *PQCrypto*, volume 5299 of *LNCS*, pages 31–46, 2008.
10. D. J. Bernstein, T. Lange, and C. Peters. Attacking and defending the McEliece cryptosystem. In *PQCrypto*, pages 31–46, 2008.
11. D. J. Bernstein, T. Lange, C. Peters, and H. van Tilborg. Explicit bounds for generic decoding algorithms for code-based cryptography. In *Pre-proceedings of WCC 2009*, pages 168–180, 2009.
12. Daniel J. Bernstein, Tanja Lange, and Christiane Peters. Smaller decoding exponents: Ball-collision decoding. In Phillip Rogaway, editor, *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 743–760. Springer, 2011.
13. Wieb Bosma, John J. Cannon, and Catherine Playoust. The Magma algebra system I: The user language. *Journal of Symbolic Computation*, 24(3-4):235–265, 1997.
14. Buchberger, B. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal.* PhD thesis, Innsbruck, 1965.
15. A. Canteaut and F. Chabaud. A new algorithm for finding minimum-weight words in a linear code: Application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. *IEEE Transactions on Information Theory*, 44(1):367–378, 1998.
16. D. A. Cox, J. B. Little, and D. O'Shea. *Ideals, Varieties, and algorithms: an Introduction to Computational Algebraic Geometry and Commutative Algebra.* Undergraduate Texts in Mathematics, Springer-Verlag, New York., 2001.
17. J.-C. Faugère. A new efficient algorithm for computing gröbner bases (F4). *Journal of Pure and Applied Algebra*, 139(1-3):61–88, 1999.
18. J.-C. Faugère. A new efficient algorithm for computing gröbner bases without reduction to zero : F5. In *ISSAC'02*, pages 75–83. ACM press, 2002.
19. Jean-Charles Faugère. FGb: A Library for Computing Grbner Bases. In Komei Fukuda, Joris Hoeven, Michael Joswig, and Nobuki Takayama, editors, *Mathematical Software - ICMS 2010*, volume 6327 of *Lecture Notes in Computer Science*, pages 84–87, Berlin, Heidelberg, September 2010. Springer Berlin / Heidelberg.
20. Jean-Charles Faugère, Valérie Gauthier, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. A distinguisher for high rate McEliece cryptosystems. *IEEE Transactions on Information Theory*, 2013. To appear.
21. Jean-Charles Faugère, Valérie Gauthier-Umana, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. A Distinguisher for High Rate McEliece Cryptosystems. In *Information Theory Workshop (ITW), 2011 IEEE*, pages 282–286, October 2011.
22. Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. Algebraic cryptanalysis of mceliece variants with compact keys. In Gilbert [26], pages 279–298.
23. Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. Algebraic Cryptanalysis of McEliece variants with compact keys – toward a complexity analysis. In *SCC '10: Proceedings of the 2nd International Conference on Symbolic Computation and Cryptography*, pages 45–55, RHUL, June 2010.
24. M. Finiasz and N. Sendrier. Security bounds for the design of code-based cryptosystems. In M. Matsui, editor, *Asiacrypt 2009*, volume 5912 of *LNCS*, pages 88–105. Springer, 2009.
25. P. Gaborit. Shorter keys for code based cryptography. In *Proceedings of the 2005 International Workshop on Coding and Cryptography (WCC 2005)*, pages 81–91, Bergen, Norway, March 2005.
26. Henri Gilbert, editor. *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*. Springer, 2010.
27. Stefan Heyse. Implementation of mceliece based on quasi-dyadic goppa codes for embedded devices. In Bo-Yin Yang, editor, *Post-Quantum Cryptography*, volume 7071 of *Lecture Notes in Computer Science*, pages 143–162. Springer Berlin Heidelberg, 2011.
28. P. J. Lee and E. F. Brickell. An observation on the security of McEliece's public-key cryptosystem. In *Advances in Cryptology - EUROCRYPT'88*, volume 330/1988 of *Lecture Notes in Computer Science*, pages 275–280. Springer, 1988.
29. J. S. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Transactions on Information Theory*, 34(5):1354–1359, 1988.
30. P. Loidreau and N. Sendrier. Weak keys in the McEliece public-key cryptosystem. *IEEE Transactions on Information Theory*, 47(3):1207–1211, 2001.
31. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Gilbert [26], pages 1–23.
32. F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes.* North–Holland, Amsterdam, fifth edition, 1986.

33. Alexander May, Alexander Meurer, and Enrico Thomae. Decoding random linear codes in $\tilde{O}(2^{0.054n})$. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT*, volume 7073 of *Lecture Notes in Computer Science*, pages 107–124. Springer, 2011.
34. R. J. McEliece. *A Public-Key System Based on Algebraic Coding Theory*, pages 114–116. Jet Propulsion Lab, 1978. DSN Progress Report 44.
35. R. Misoczki and P. S. L. M. Barreto. Compact McEliece keys from Goppa codes. In *Selected Areas in Cryptography (SAC 2009)*, Calgary, Canada, August 13-14 2009.
36. Rafael Misoczki and Paulo S. L. M. Barreto. Compact mceliece keys from goppa codes. *IACR Cryptology ePrint Archive*, 2009:187, 2009.
37. N. Patterson. The algebraic decoding of Goppa codes. *IEEE Transactions on Information Theory*, 21(2):203–207, 1975.
38. Edoardo Persichetti. Compact mceliece keys based on quasi-dyadic srivastava codes. *J. Mathematical Cryptology*, 6(2):149–169, 2012.
39. Christiane Peters. Information-set decoding for linear codes over f$_q$. In Nicolas Sendrier, editor, *PQCrypto*, volume 6061 of *Lecture Notes in Computer Science*, pages 81–94. Springer, 2010.
40. N. Sendrier. Finding the permutation between equivalent linear codes: The support splitting algorithm. *IEEE Transactions on Information Theory*, 46(4):1193–1203, 2000.
41. Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 617–635. Springer, 2009.
42. J. Stern. A method for finding codewords of small weight. In G. D. Cohen and J. Wolfmann, editors, *Coding Theory and Applications*, volume 388 of *Lecture Notes in Computer Science*, pages 106–113. Springer, 1988.
43. V. Gauthier Umana and G. Leander. Practical key recovery attacks on two McEliece variants. In *International Conference on Symbolic Computation and Cryptography–SCC*, volume 2010, page 62, 2010.

A Proof of Theorem 5

We recall that a central tool in the construction of QM codes is the sequence of $\mathbb{F}_p$-independent elements $\alpha_0, \ldots, \alpha_{\lambda-1} \in \mathbb{F}_{q^m}$ (Theorem 3). They generate a group G of size p^λ whose elements are defined for $\ell \in [0, \ldots, p^\lambda - 1]$ by $g_\ell = \sum_{j=0}^{\lambda-1} \ell_j \alpha_j$, $\ell = \sum_{j=0}^{\lambda-1} \ell_j p^j$ the decomposition in base p of ℓ. The support is chosen so as to satisfy Equation (4). An example of resolution of (4) is in [35, Algorithm 1] and [3, Algorithm 3] (by setting $\alpha_i = h_{a_i}^{-1} + \omega$). Then, the Goppa polynomial is derived from its roots A that are picked in G. The case $t = p^\lambda$ corresponds to a Goppa polynomial $\Gamma(z)$ whose roots A are all the elements of G. The fact that its roots form a group is the crucial point for proving Equation (5), and to exhibit the structure of the automorphism group. The cases where $t \neq p^\lambda$ are those when A is a sub-set of G (and not a sub-group). The associated Goppa code is not stable by the expected permutations σ_ℓ (as defined in Proposition 1), so folding the code is not possible any more. So, the idea is to select a subspace of the codewords that is stable by more permutations. This is formalized by the following statement:

Lemma 1. *Let $\alpha_0, \ldots, \alpha_{\lambda-1} \in \mathbb{F}_{q^m}$ as in Theorem 2, generating $G = \{g_0, \ldots, g_{p^\lambda-1}\}$. Let $A = \{g_0, \ldots, g_{t-1}\}$ be a subset of G generating G Let $\boldsymbol{x}$ be built thanks to the α_i's according to (4). As a consequence, each $g \in G$ corresponds to an offset $0 \leqslant j_g < p^\lambda$ of the indices on the support such that $x_{i \oplus j_g} = x_i + g$ for all $i, 0 \leqslant i \leqslant n-1$. The public code is defined by $\mathscr{C}_{pub}^{(t)} = \mathscr{G}\big(\boldsymbol{x}, \prod_{g \in A}(z-g)\big)$, and we define $\mathscr{C}^{(p^\lambda)} = \mathscr{G}\big(\boldsymbol{x}, \prod_{g \in G}(z-g)\big)$. It holds that:*

$$\boldsymbol{c} \in \mathscr{C}^{(p^\lambda)} \iff \forall g \in G, \boldsymbol{c}^{\sigma_{j_g}} \in \mathscr{C}_{pub}^{(t)}.$$

Proof. Let $g \in G$. Recall the relation $P_{\boldsymbol{c}^{\sigma_{j_g}},\boldsymbol{x}}(z) = P_{\boldsymbol{c},\boldsymbol{x}}(z-g)$, proven in the proof of Proposition 2. We have

$$\begin{aligned}
\boldsymbol{c}^{\sigma_{j_g}} \in \mathscr{C}_{pub}^{(t)} &\iff \prod_{a\in A}(z-a) | P_{\boldsymbol{c}^{\sigma_{j_g}},\boldsymbol{x}}(z) \\
&\iff \prod_{a\in A}(z-a) | P_{\boldsymbol{c},\boldsymbol{x}}(z-g) \\
&\iff \prod_{a\in A}(z+g-a) | P_{\boldsymbol{c},\boldsymbol{x}}(z) \\
&\iff \prod_{a\in A-g}(z-a) | P_{\boldsymbol{c},\boldsymbol{x}}(z)
\end{aligned}$$

As all the elements of G are pairwise distinct, the polynomials $(z-a)$ are coprime. The least common multiple of all the polynomials $\prod_{a\in A-g}(z-a)$ is $P = \prod_{g\in G}(z-g)$. So, we conclude,

$$\begin{aligned}
\forall g \in G, \boldsymbol{c}^{\sigma_{j_g}} \in \mathscr{C}_{pub}^{(t)} &\iff \forall g \in G \prod_{A\in A-g}(z-g) | P_{\boldsymbol{c},\boldsymbol{x}}(z) \\
&\iff \textstyle\prod_{g\in G}(z-g) | P_{\boldsymbol{c},\boldsymbol{x}}(z) \\
&\iff \boldsymbol{c} \in \mathscr{C}^{(p^\lambda)}.
\end{aligned}$$

The lemma permits to deduce a parity-check matrix of $\mathscr{C}^{(p^\lambda)}$ from any parity-check matrix $\boldsymbol{H}$ of $\mathscr{C}_{pub}^{(t)}$. Indeed, observe that for any row $\mathbf{h}$ of $\boldsymbol{H}$ and any permutation σ of the indices:

$$\begin{aligned}
\boldsymbol{c}^{\sigma} \in \mathscr{C}_{pub}^{(t)} &\iff \textstyle\sum_{i=0}^{n-1} c_{\sigma(i)} h_i = 0 \\
&\iff \textstyle\sum_{i=0}^{n-1} c_i h_{\sigma^{-1}(i)} = 0
\end{aligned}$$

To ensure that a word $\boldsymbol{c}$ and all its permuted words $c^{\sigma_{j_g}}$ for $g \in G$ belong to $\mathscr{C}_{pub}^{(t)}$, it suffices to permute the rows of $\boldsymbol{H}$ according to all the σ_g's with $g \in G$ and concatenate the obtained matrices. This is precisely what the function Δ_{p^λ} does for a group G of size p^λ.